Orchestrating caGrid services in Taverna

Wei Tan[1], Ravi Madduri[1, 2], Kiran Keshav[3], Baris E. Suzek[4], and Scott Oster[5], Ian Foster[1]

[1]Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL, USA [2]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA

[3]Center for Computational Biology, Columbia University [4]Georgetown University Medical Center, Department of Biochemistry and Molecular & Cellular Biology [5]Department of Biomedical Informatics, Ohio State University

OVERVIEW

For the empowerment of users from biological or medical domains in creating and executing their workflows efficiently, the caGrid Workflow team, with the ICR working group, has selected the Taverna workbench and successfully created a prototype to orchestrate caGrid Data and Analytical services for ICR workflows. This prototype is the first step towards achieving our goal of providing an easy-to-use workflow authoring and submission tool that will be capable of orchestrating caGrid data and analytical services in executing workflows. Now, we commit ourselves to provide caGrid Workflow builder and Workflow Service as a tool which will eventually support caBIG users across workspaces in creating and executing their domain based workflows.

Web Resources:

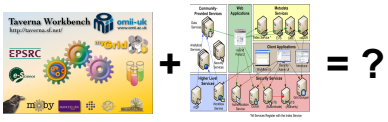
Taverna: http://taverna.sourceforge.net/

GT4 Plug-in download: http://www-unix.mcs.anl.gov/~madduri/taverna/

caBIG: http://www.cagrid.org/mwiki/index.php?title=CaGrid

Create CaGrid Workflow Using Taverna : http://www.cagrid.org/wiki/CaGrid:How-To:Create_CaGrid_Workflow_Using_Taverna

WHY TAVERNA?



caGrid

- A primary goal of caBIG is to integrate a wide variety of distributed, cancer-related data and analytic resources. This goal is achieved by virtualizing these resources via grid services.
- Inevitably, users then want to interconnect multiple such services to perform more complex data collection and analysis tasks. The resulting analytical workflows typically involve the consumption of both analytic and data services. Data is transformed and piped between services, and logical decisions are made along the way.
- This type of workflow is data-flow oriented, as opposed to the control-flow oriented business process in which a workflow refers to the steps that a business takes through processing.
- This need to support data-flow-oriented workflows was especially relevant when deciding on a solution for orchestrating workflows in caGrid.

Features that align with the caGrid requirements are:

•Explicit modeling of data flow:
In Scufl (the workflow language that Taverna adopts), data are passed between processors in a data-flow style. There is no process-level data definition, and data items are passed by default in a processor-to-processor manner. Scufl's compact representation makes it an ideal choice for modeling caGrid workflows that usually involve data pipelines among analytical and data services.

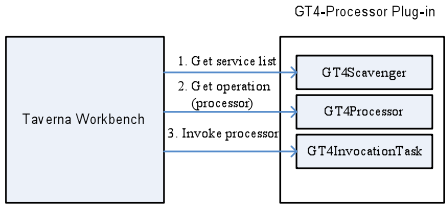
•Implicit iteration:
In Scufl, implicit iteration occurs if a processor receives more inputs than it expects. This capability is useful when the cardinality of inputs cannot be estimated at build-time.

•Input/Output metadata:
A processor's input and output data types can be tagged with ontology terms from the myGrid ontology, with an arbitrary text description, and/or with MIME types. This semantic metadata mechanism provides a feasible integration point with caGrid metadata and discovery services.

•Beanshell scripting and XML processing support:
Taverna allows for the definition of custom processors that execute beanshell scripts (a flavor of dynamic Java, see http://www.beanshell.org/) inside a Taverna workflow.

•Easy to use functionality:
Taverna provides both build-time and run-time support for workflow modeling, debugging, tracing, and provenance. Furthermore, its intuitive user interface allows even non-IT specialists to use these functionalities. Thus, Taverna is well suited for caGrid users from biological or medical domains who may not have strong IT expertise.

DESIGN

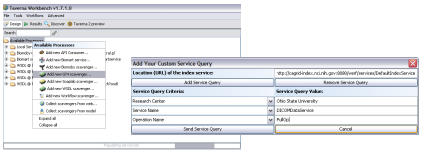


The interaction between Taverna Workbench and GT4-Processor plug-in

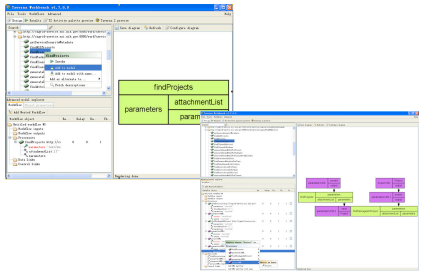
- We developed a plug-in called GT4-Processor, to let Taverna access the caGrid services.
- The figure on the left illustrates the interaction between the Taverna workbench and the GT4-Processor plug-in.
  - First, in order to discover registered caGrid services with semantic metadata, we integrate caGrid discovery API in the GT4 plug-in's scavenger (a Taverna extension point).
  - After the workbench retrieves a list of caGrid services and their operations, users can select among these operations and add them to workflows.
  - At runtime when the workflow execution reaches a GT4 processor, the engine fetches the service's metadata (from the workflow definition file), the input data (from the workbench), wraps the data into a SOAP request package, sends the package to the target EPR, and finally retrieves the output data.

IMPLEMENTATION

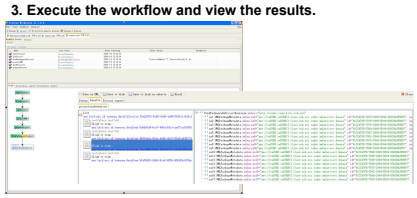
1. Semantic/metadata based service discovery.



2. Build a workflow using the services obtained by discovery.

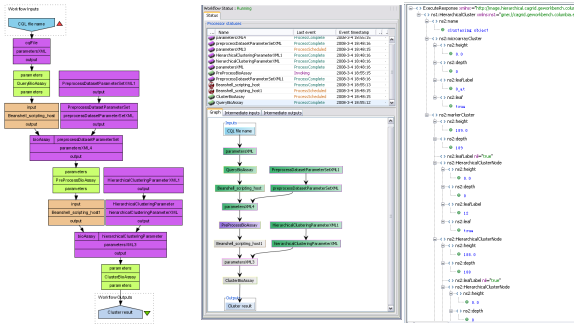


3. Execute the workflow and view the results.



SAMPLE WORKFLOW AND EXECUTION

- To illustrate the application of the GT4-Processor plug-in, we describe its use to construct a microarray data analysis workflow that invokes caGrid services hosted at multiple institutions. This workflow contains three major steps:
  1. Querying and retrieving the microarray data (bioassays) of interest from a caArray data service hosted at Columbia University.
  2. Preprocessing the microarray data using the GenePattern analytical service hosted at the Broad Institute at MIT.
  3. Running hierarchical clustering on the preprocessed data using the geWorkbench analytical service hosted at Columbia University.
- The Taverna workflow (shown in leftmost figure) contains an input processor, an output processor, three GT4 processors representing three caGrid services, and other "shim" processors like xml splitters and beanshell scripts to deal with data transformation between services.
- Rightmost figure shows the execution trace and execution result, i.e., the clustered arrays.



The sample caGrid workflow, its execution trace and result